A Simple Password Authentication Scheme Based on Geometric Hashing Function

Xu Zhuang[1], Chin-Chen Chang[2,3], Zhi-Hui Wang[4], and Yan Zhu[5]
(Corresponding author: Zhi-Hui Wang)

School of Information Science and Technology, Southwest Jiaotong University[1]
Chengdu, Sichuan, China
Department of Information Engineering and Computer Science, Feng Chia University[2]
Taichung City 40724, Taiwan
Department of Computer Science and Information Engineering, Asia University[3]
Taichung 41354, Taiwan
School of Software, Dalian University of Technology, Dalian, Liaoning, China[4]
School of Information Science and Technology, Southwest Jiaotong University[5]
Chengdu, Sichuan, China
(Email: wangzhihui1017@gmail.com)

(Received Jan. 26, 2013; revised and accepted May 15, 2013)

Abstract

Password authentication protocol is one of most important mechanisms to prevent resources from accessing by unauthorized users. Many password authentication schemes have been developed in last decades, and many of them are based on the use of smart card. However, nowadays it is not applicable for many applications on the Internet to adopt the smart card in their authentication scheme due to its inconvenience and relative high cost for both users and service providers. On the other hand, all password authentication schemes without using smart card (*ex*. schemes based on hash function) cited in this paper are proved to be vulnerable to replay attack and/or man-in-the-middle attack and/or verifier stolen attack, *etc*. Hence, in this paper, we focus on designing a simple, secure and high efficiency password authentication scheme based on geometric hash function without using smart card. Our security analysis and performance evaluation demonstrate that our scheme is quite simple and efficient, and also can withstand replay attack, password guessing attack, min-in-the-middle attack, verifier stolen attack and denial-of-service attack.

Keywords: Geometric hash function, hash table, one-way hash function, password authentication

1 Introduction

In many insecure transmission environments, such as Internet and wireless channel, when a user wants to access the valuable recourse in the server, typically, a user's identity *id* and password *pw* are needed for the server to authenticate the validity of the user. Password authentication is one of the best-known and simplest mechanisms for dealing with the transmission of secret data of a login system over insecure networks [21].

Traditional password authentication schemes store a verifier table for user authentication [8, 11, 12, 15, 16, 18]. In these schemes, generally, the user uses his/her identity id and password pw to login the system. In most cases, the identity id is considered as public, while the password pw is a secret. Many applications over Internet adopt this technique due to its easy to implement and relative low cost.

Recent decades, as the development of the smart cards, a massive of smart card based password authentication schemes have been proposed [2, 3, 14, 17, 19, 20, 22]. Many of these schemes are verifier free, which means that the server doesn't keep a verifier table for user authentication. However, each user in these systems needs to hold a smart card. Actually, these smart cards must be kept very carefully, because an attacker may compromise theses cards and steal all secrets in them, which could lead lots of security concerns. Besides that, compared with the traditional solutions, these schemes may increase the cost of both the users and the service providers due to the extra smart card.

In the real environment, it is not impossible for many applications to adopt smart cards due to its poor flexibility, especially for applications over Internet in which the service provider need to take into account the additional implementation cost [1]. Thus, in this paper, we focus on the group of schemes based on the use of verifier table without using smart card.

A straightforward method to ensure the data transmitted via an insecure channel is to adopt conventional encryption algorithm (such as RSA, DES and etc.). However, the heavy computation load on both the end-user and the server

makes it is not suitable for real-time data exchange. Thus, many researchers began to focus on designing more lightweight password authentication schemes.

In 1981, Lamport [11] proposed a password authentication scheme based on one-way hash function to guarantee the secrecy of the password table in the server [7]. After that, an increasing amount of researchers proposed schemes [2, 3, 5, 9, 12, 15, 16, 18] to withstand various attacks, including off-line password guessing attacks, replay attacks, denial of service attacks, stolen-verifier attacks, and so on [21]. In 2000, Peyavian and Zunic [16] proposed a remote user authentication scheme based on hash function; however, since an intruder can easily get the user's identity id and two random numbers transmitted over an insecure channel, the intruder can obtain the user's password by off-line guessing. To overcome this drawback, Lee et al. [12] developed an improved scheme based on the work by Peyavian and Zunic [16]; however, as reported in [9], their scheme is also vulnerable to off-line guessing attack, denial-of-service attack and stolen-verifier attack. Sandirigama and Shimizu [18] proposed a efficient password authentication scheme named SAS (Simple And Secure password authentication). Unfortunately, Lin et al. [15] presented a replay attack and a DoS attack against SAS. They also proposed an improvement OSPA (Optimal Strong-Password Authentication) to repair the thoughtless designs of SAS. However, OSPA is also vulnerable to replay attack, stolen-verifier attack, DoS attack and guessing attack [4, 10, 13]. Later, Ku [8] developed a hash based password authentication scheme, which is proved to be vulnerable to the min-in-the-middle attack by Yang and Shen [23].

In this paper, we propose a very efficient password authentication scheme based on a geometric hashing without using smart card. Our scheme has the ability to withstand replay attack, password guessing attack, min-in-the-middle attack, verifier-stolen attack and denial-of-service attack. Our scheme has another merit that the user's identity id never occurs in the insecure channel as a plain text, which can enhance the security of the protocol and protect the user's anonymity. The rest of the paper is organized as follows: Section 2 introduces the geometric hash function on which our scheme is based. In Section 3, our scheme is proposed. Section 4 provides security analysis and performance evaluation. The conclusions are drawn in Section 5.

2 Related Work

Comer and O'Donnell [6] proposed an efficient algorithm to produce a perfect hashing function of a set of keys in two-dimensional space. In this section, we will first give definitions related to the geometric problem and then describe their main idea about it. More details would not be discussed in this paper, because all concepts introduced in this section are enough for clarifying our scheme in later sections.

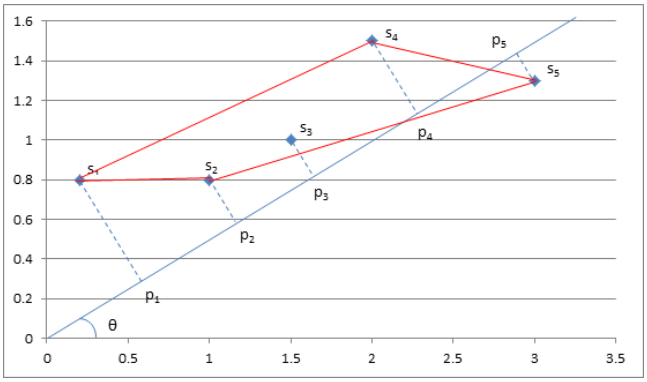

Figure 1: Example for introducing the definitions

Definitions and notations:

S: a set of *n* points in two-dimensional space, $S = \{(x_i, y_i) \mid x_i, y_i \in R, 1 \le i \le n\}$.

θ: called angle of projection, $\theta \in [0, \pi)$. In two dimensional space, each angle corresponds to countless lines. For convenience, we always choose the one across the original point in computations through this paper.

$|\, s_i \,|_\theta$: The projection of point s_i at angle θ. For $s_i = (x, y)$, the projection at angle θ is

$$|\, s_i \,|_\theta = x \cdot \cos\theta + y \cdot \sin\theta . \quad (1)$$

P_θ^S: A set of all projections of pairs in *S* at angle θ,

$$P_\theta^S = \{|\, s_i \,|_\theta \mid s_i \in S\} . \quad (2)$$

Usually, if *S* is known, we use notation P_θ instead of P_θ^S.

$span(P_\theta)$: The span of P_θ,

$$span(P_\theta) = \max(\left||\, s_i \,|_\theta - |\, s_j \,|_\theta\right|), s_i, s_j \in S . \quad (3)$$

$res(P_\theta)$: The resolution of projection P_θ,

$$res(P_\theta) = \min(\left||\, s_i \,|_\theta - |\, s_j \,|_\theta\right|), i \ne j \text{ and } s_i, s_j \in S . \quad (4)$$

$len(P_\theta)$: The length of projection P_θ,

$$len(P_\theta) = \frac{span(P_\theta)}{res(P_\theta)} . \quad (5)$$

The geometric problem: find the angle of projection $\theta \in [0, \pi)$ which minimizes $len(P_\theta)$ for a given *S*, a set of *n* points in two-dimensional space. For convenience, we always use the notation *GP* to represent this geometric problem in the rest of the paper.

Figure 1 gives an example to clarify aforementioned definitions. In Figure 1, $S = \{s_1, s_2, s_3, s_4, s_5\}$, and each point's ($s_i$) projection is denoted as p_i. Obviously, because the distance between p_1 and p_5 is longest, $span(P_\theta) = \left||\, s_1 \,|_\theta - |\, s_5 \,|_\theta\right|$, and because the distance between p_2 and p_3 is shortest $res(P_\theta) = \left||\, s_2 \,|_\theta - |\, s_3 \,|_\theta\right|$. Note that, the value of p_i is actually equals the distance between the original point and p_i. So, in Figure 1, the *GP* is to find an

angle of projection θ, which leads a minimal value of $len(P_\theta) = span(P_\theta)/res(P_\theta) = \left(\left|s_1\right|_\theta - \left|s_5\right|_\theta\right) / \left(\left|s_2\right|_\theta - \left|s_3\right|_\theta\right)$.

Index	Hashing value
0	$P(\left\|(id_0, pw_0)\right\|_\theta)$
1	null
2	$P(\left\|(id_1, pw_1)\right\|_\theta)$
……	……
n-1	null
n	$P(\left\|(id_n, pw_n)\right\|_\theta)$

$P(\left\|(id_3, pw_3)\right\|_\theta)$

Figure 2: Example of the hash table

Obvously, we can change θ from 0 to π to compute $span(P_\theta)$, $res(P_\theta)$ and then $len(P_\theta)$ for each θ. Then, we find the expected θ which produces the minimum $len(P_\theta)$ among all computed $len(P_\theta)$'s. To reduce the time complexity, Comer and O'Donnell [6] offer an efficient way to solve the *GP*. Their algorithm presents a solution with time complexity $O(n^2 \log n)$ [6]. For more details about their algorithm, please refer to the literature [6].

3 Proposed Scheme

In this section, we shall propose our scheme based on the *GP* introduced in Section 2. First, we give an overview of our scheme and then discuss the most important phases: registration phase, login phase and authentication phase as well as password changing phase.

In our password authentication scheme, the server holds a hash table which keeps a hash value for each record, as shown in Figure 2. If we consider each user's identity id_i and password pw_i as a point (id_i, pw_i) in two dimensional space, we can use the method introduced in [6] to compute the angle of projection θ which minimize $len(P_\theta)$ for set $S_{id\text{-}pw}$, where $S_{id\text{-}pw}$ consists of users' (id_i, pw_i) pairs. Obviously, that is exactly the *GP* problem. Using the method described in Section 2, we can compute the expected angle of projection and resolution for set $S_{id\text{-}pw}$.

Assume that we have computed the expected angle of projection θ^* and the resolution *res* for $S_{id\text{-}pw}$. For a given pair $s_i = (id_i, pw_i)$ in $S_{id\text{-}pw}$, we first compute its projection as follow:

$$P(\left|s_i\right|_\theta) = P(\left|(id_i, pw_i)\right|_\theta) = \cos\theta \cdot id_i + \sin\theta \cdot pw_i + C \quad (6)$$

In Formula 6, the parameter C is a constant used to adjust the minimal projection of $P_\theta^{S_{id\text{-}pw}}$ to zero. This would be helpful to lead the hash table begin with index 0 in our later work.

$$C = -\min(\{P(\left|(id_j, pw_j)\right|_\theta) \mid P(\left|(id_j, pw_j)\right|_\theta \in P_\theta^{S_{id-pw}}\}) \quad (7)$$

Next, we can compute the index of pair $s_i = (id_i, pw_i)$ in the hash table by following formula:

$$Index(s_i) = Index(id_i, pw_i) = \left\lfloor P(\left|s_i\right|_\theta) / res(P_\theta^{S_{id-pw}}) \right\rfloor. \quad (8)$$

After above work, given a pair $s_i = (id_i, pw_i)$ in $S_{id\text{-}pw}$, we can adds its projection $P(|s_i|)_\theta$ (hash value) into the hash table with index $Index(s_i)$.

Three points worth noting here:

1) Because the resolution *res* is the minimal distance between any two different projections in $P_\theta^{S_{id\text{-}pw}}$, Formula (8) guarantee that there will never be any two different pairs which have the same index, which means that our hash function is perfect (no conflict). On the other hand, the size of the hash table equals $\left\lfloor len(P_\theta^{S_{id-pw}}) \right\rfloor + 1$. In Section 2, our task is to find a θ which minimizes $len(P_\theta^S)$ for set *S*. Now, we can find that the *GP* actually aims to find a minimal size and perfect hash table for a given set consisting of two dimensional elements.
2) Thanks to the constant *C* used to adjust the minimal projection of $P_\theta^{S_{id\text{-}pw}}$ to 0, the minimal index of the hash table is adjusted to 0.
3) Generally, each record just has one hash value. But for easy to maintenance the system, we attach a linked list to the record which is necessary to store more values. This case would happen in registration phase and password changing phase. More about this issue will be discussed in Subsection 3.1.

After creating the hash table, the server deletes the set $S_{id\text{-}pw}$ and publishes the angle of projection θ^*, resolution *res* and constant *C*. Note that, once the server has computed such a hash table, the set $S_{id\text{-}pw}$ will be never used in future work. So, initially, our scheme needs to collect several (>= 2, theoretically) *id-pw* pairs for creating the hash table. For easy to describe all phases of our scheme, we assume that the server has computed such a hash table and published θ^*, *res* and *C*.

3.1 Registration Phase

The registration phase is invoked whenever a user wants to apply for a new account to join the system. The user first freely chooses his/her identity *id* and password *pw*, and then sends *id* and *pw* to the server via a secure channel. Upon receiving the registration request with pair (*id*, *pw*), the server does the following steps to respond it:

R1: The server uses Formula (6) to compute the projection P_{new} of pair (*id*, *pw*), and then uses Formula (8) to compute its index i_{new}.

R2: If i_{new} is bigger than or equal to the table size, the server extends the original table to size i_{new}+1 (i_{new} is the last index), and adds P_{new} to the record with index i_{new}. Otherwise, server locates the record with index i_{new} and checks whether it is null. If it is null, server adds P_{new} to it directly; otherwise, two cases should be considered:

Case 1: If P_{new} is not equal to all projections in this record, the server attaches a list node with value P_{new} to this record.

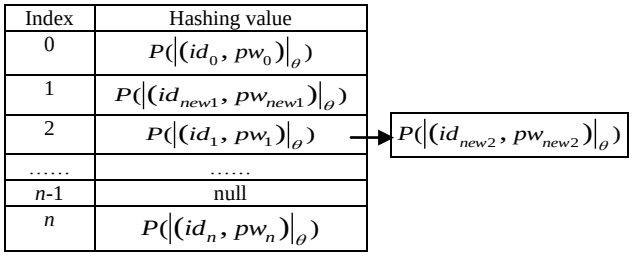

Figure 3: Example of registration

Case 2: If P_{new} is equal to one of these projections in the record, the server rejects the registration request and prompts the user to choose a new identity *id*.

In our scheme, a very significant difference compared with other schemes is that, different users may have the same identity *id*, but never the same $P(|id, pw|_\theta)$. Figure 3 is an example of registration. In Figure 3, there are two new users who choose their identities and passwords as (id_{new1}, pw_{new1}) and (id_{new2}, pw_{new2}), respectively. The server then computes their projections and indexes by Formula (6) and (8). Assuming $Index(id_{new1}, pw_{new1}) = 1$ and $Index(id_{new2}, pw_{new2}) = 2$, since the hashing value of index 1 is null, server adds $P(|(id_{new1}, pw_{new1})|_\theta)$ to it. Due to the value of index 2 is not null and we assume that $P(|(id_1, pw_1)|_\theta) \neq P(|(id_{new2}, pw_{new2})|_\theta)$, server attaches a link list to it.

3.2 Login Phase

When a user wants to login the system, this phase will be first carried out.

L1: The user submits his/her identity id_r and password pw_r to a client.

L2: Upon reviving the request with pair (id_r, pw_r), the client computes $m_1 = P(|(id_r, pw_r)|_\theta)$, $m_2 = h(pw_r \oplus id_r)$, $M_1 = Index(|(id_r, pw_r)|_\theta)$, $M_2 = h(m_1 \oplus t_c)$, $M_3 = pw_r \oplus m_1$, $M_4 = P(|(m_2, pw_r)|_\theta)$, where h(·) denotes a one-way hash function and t_c is the time stamp. Then, the client sends the authentication messages M_1, M_2, M_3, M_4 and t_c to server.

3.3 Authentication Phase

The authentication phase is invoked when the server gets an authentication request.

A1: The server checks whether $0 < t_s - t_c < t_r$, where t_s is the current time stamp of the server and t_r is a reasonable time delay threshold. If it holds, next step is carried out; otherwise, the server rejects the request.

A2: If M_1 is bigger than or equal to the table size, server rejects the request. Otherwise, the server uses the M_1 as an index to locate the corresponding record in the hash table. If the value with index M_1 is null, the server terminates current session; otherwise, the server uses each hash value $P(|(id_i, pw_i)|_\theta)$ in the record to compute $M_2' = h(P(|(id_i, pw_i)|_\theta) \oplus t_c)$. If one of these computed M_2's (assume by using hash value P_m) equals the received M_2, next step will be carried out; otherwise, the server terminates the current session.

A3: The server uses P_m to extract pw_e from M_3 by $M_3 \oplus P_m$. Then the server uses pw_e and p_m can get the id_e utilizing Formula (6). After that, the server can compute a local $m_2' = h(pw_e \oplus id_e)$ and $M_4' = P(|(m_2', pw_e)|_\theta)$. If M_4' equals the received one, the server approve the user's login request.

3.4 Password Changing Phase

When a user wants to change his password, he first needs to login the system using his original identity id_o and password pw_o. If the user login the system successfully, then, following steps are performed.

C1: The user submits his new password pw_n to the client. The client then computes $m = P(|(id_o, pw_n)|_\theta)$ and transmits messages $M_{C1} = m \oplus P(|(id_o, pw_o)|_\theta)$ and $M_{C2} = h(m) \oplus h(P(|(id_o, pw_o)|_\theta))$ to the server.

C2: Due to the pre-performed login and authentication phases, the server can note down the $P(|(id_o, pw_o)|_\theta)$ once the user successfully login the system. After receiving the message M_{C1}, the server can extract the *m*' by doing $M_{C1} \oplus P(|(id_o, pw_o)|_\theta)$. And then the server computes $h(P(|(id_o, pw_o)|_\theta))$ to extract $h(m)$ from M_{C2}. Finally, the server uses the extracted *m*' to compute a local $h(m')$ and check whether $h(m) = h(m')$. If it holds, next step will be carried out; otherwise, the server rejects the password changing request.

C3: The server uses Formula (8) to compute the new index of the user using the extracted $P(|(id_o, pw_n)|_\theta)$. And handle this new index as the same way as the registration phase.

4 Security Analysis and Performance Evaluation

Next, we will discuss the security of our scheme against various kinds of attacks and its performance efficiency. First of all, there is a very different point of our scheme compared with others. That is the user identity *id* is also can be considered as a secret of the user because it is never transmitted as plain text in our scheme. But the security of our scheme is not depended on this assumption.

4.1 Security Analysis

In this subsection, we will discuss the ability of our proposed scheme to withstand replay attack, off-line guessing attack, stolen-verifier attack, denial-of-service attack and min-in-the middle attack.

Resistance to replay attack

An attacker can intercept messages M_1, M_2, M_3, M_4 and t_c of a user's authentication messages in last session over an insecure channel. Because a time stamp is used in M_2, an attacker cannot replay all messages with the intercepted t_c directly. This attack can be detected by the server easily because $t_s - t_c > t_r$. So, the attacker may tries to changes t_c

Table 1: Comparisons among our scheme and othersAssume each string used in these protocols has length L. In our scheme, the total number of strong one way hash function used depends on the current record used to authenticate the user., so that n represents the size of linked list attached to the current record.

	SAS [18]	OSPA [15]	Peyravian and Zunic's Scheme[16]	Lee-Li-Hwang's scheme[12]	Ku's scheme[8]	Our scheme
Resistance to replay attack	no	no	no	no	no	yes
Resistance to guessing attack	no	no	no	no	no	yes
Resistance to verifier-stolen attack	no	no	no	no	no	yes
Resistance to min-in-the-middle attack	no	no	no	no	no	yes
Resistance to Dos attack	no	no	no	no	no	yes
Storage space for each user in server	3L	3L	2L	2L	3L	1L
Total messages transmitted in a single session	4	4	5	5	6	4
Total rounds in a single session	3	3	3	3	3	1
Total number of strong one way hash function used	8	11	3	3	12	4+n

to a closest time stamp $t_{c'}$. However, it is also useless because the server will compute a mismatched M_2' using a changed time stamp from the client, which leads the server reject the user's login request.

Resistance to off-line guessing attack

As the same assumption made in Subsection 4.1, the attacker has stolen the authentication messages of a user, M_1, M_2, M_3, M_4 and t_c. Although an attacker can utilize Formula (8) to compute the $\lfloor P(|(id, pw)|_\theta) \rfloor$ by doing $M_1 \times res$, the attacker still cannot get the user's password because the attacker doesn't know the user's identity *id*. Even if the attacker get the user's *id* by some other ways, it is impossible for him to guess the password of the user because there are a lot of candidates *pw'* which satisfy the equation $\lfloor P(|(id, pw)|_\theta) \rfloor = \lfloor P(|(id, pw')|_\theta) \rfloor$.

Resistance to stolen-verifier attack

The stolen-verifier attack means that if an attacker has stolen a verifier stored in the server, then s/he can use it directly to masquerade as an authorized user to login the system. In the proposed scheme, if an attacker has stolen a verifier stored in the hash table, he still cannot login the system. With the stolen verifier P_s, the attacker can forge messages M_1 and M_2. However, due to the attacker doesn't know the user's password *pw* (the attacker also doesn't know the user's *id* just from the transmitted messages), he cannot forge messages M_3 and M_4. Therefore, it is very difficult for the attacker to login the system without any information about the user's password.

Resistance to denial-of-service (DoS) attack

This attack generally occurs in the verifier updating and password changing phase. The one-time password authentication scheme needs a message from the user to invoke the verifier updating in the server side. However, because our scheme needn't to update user's verifier after each successful authentication, DoS attack is fully resisted in this situation. In the password changing phase, since a message M_{c2} used to ensure the data integrity, any modification of M_{c1} will be detected by the server. So, our scheme can withstand DoS attack.

Resistance to man-in-the-middle attack

A man-in-the middle attack intercepts messages both from the server and user, and then sends opposite side the forged messages to deceive the server or user. The success rate of this attack is based on the data integrity protection mechanism used in protocol. In our scheme, messages M_3 and M_4 are used to keep the data integrity in the protocol, so that any modification of messages will be detected in the server side.

4.2 Performance Evaluation

Under a perfect situation, our scheme can authenticate a user with time complexity $O(1)$. However, due to the linked list attached to the hash table, the time complexity is increased to $O(n)$, where n is the size of the longest linked list attached to the hash table. To maintenance the performance of the system, when the size of the longest linked list is larger than a given threshold, the server should to re-compute such a hash table. Fortunately, it is a quite easy task for the server because the server has stored all projections (the verifiers) of all users. The server can find out the new resolution using Formula (4) based on all

projections in the hash table, and uses Formula (8) to calculate each projection's new index.

As mentioned in Section 3, because the user's identity never occurs in the channel as plain text, the user's identity *id* also can be considered as a secret in our scheme. But the security of our scheme is not depended on this assumption. From the security analysis made above, even though a user publishes his/her identity *id*, an attacker also cannot get any clue about the user's password from the transmitted messages M_1, M_2, M_3 and M_4. Thus, in our scheme, if a user can carefully keep his identity *id*, this could make the login phase become more secure. Due to the same reason, our scheme is user anonymous.

Table 1 is a comprehensive comparison of our scheme and others'. In Ku's scheme [8], because an attacker can guess a user's password by off-line manner, it is easy for the adversary to implement other attacks after getting the right password. From Table 1, it is obvious that our scheme is quite efficient in views of space cost and total rounds needed in a single session. The total number of strong one way hash function used in our scheme is 4+*n*, where *n* is the size of linked list attached to the current record. Fortunately, as we discuses above, it is quite convenience for the server to re-compute a new hash table which is conflict resistance, so, the *n* equals 0 in most cases.

5 Conclusions

In this paper, we have proposed a password authentication scheme based on a geometric hashing problem, without using smart card. The scheme emphasizes on the properties of simplicity, practicality and high efficiency. Due to the low computation cost in the login and authentication phase, our scheme can be used in the resource-limited environment where fast password authentication is needed. As the security analysis and performance evaluation shown in Section 4, our scheme is more efficient and secure than other schemes cited in Table 1.

Acknowledgements

This work was supported by the National Nature Science Foundation of China under Grant No. 61272374.

References

[1] K. Altinkemer and T. Wang, "Cost and benefit analysis of authentication systems," *Decision Support Systems*, vol. 51, pp. 394-404, 2011.

[2] A. K. Awasthi and S. Lal, "A remote user authentication scheme using smart cards with forward secrecy," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 1246-1248, 2003.

[3] C. C. Chang and T. C. Wu, "Remote password authentication with smart cards," *IEE Proceeding-E*, vol. 138, pp. 165-168, 1991.

[4] C. M. Chen and W. C. Ku, "Stolen-verifier attack on two new strong-password authentication protocols," *IEICE Transactions on Communications*, vol. 85, no.11, pp. 2519-2521, 2002.

[5] H. R. Chung, W. C. Ku and M. J. Tsaur. Weaknesses and improvement of Wang et al.'s remote user password authentication scheme for resource-limited environment, *Computer Standards and Interfaces*, vol. 34, no. 4, pp. 863-868, 2009.

[6] D. Comer and M. J. O'Donnell, "Geometric problems with application to hashing," *SIAMJ. Comput*., vol. 11, no. 2, pp. 217-226, 1982.

[7] T. Hwang, Y. Chen and C. S. Laih, "Non-interactive password authentications without password tables," in *IEEE Region 10 Conference on Computer and Communication System*, pp. 429-431, 1990.

[8] W. C. Ku, "A hash-based strong password authentication scheme without using smart cards," *ACM Operating System Review*, vol. 38, no. 1, pp. 29-34, 2004.

[9] W. C. Ku, C. M. Chen and H. L. Lee, "Weaknesses of Lee-Li-Hwang's hash-based password authentication scheme," *ACM Operating Systems Review*, vol. 37, no. 4, pp. 19-25, 2003.

[10] W. C. Ku, H. C. Tasi and S. M. Chen, "Two simple attacks on Lin-Shen-Hwang's strong password authentication protocols," *ACM Operating Systems Review*, vol. 37, no. 4, pp. 26-31, 2003.

[11] Lamport, "Password authentication with insecure communication," *Commmunication of the ACM*, vol. 24, no. 11, pp. 770-772, 1981.

[12] C. C. Lee, L. H. Li and M. S. Hwang, "A remote user authentication scheme using hash functions," *ACM Operating System Review*, vol. 36, no. 4, pp. 23-29, 2002.

[13] C. C. Lee, C. H. Liu and M. S. Hwang, "Guessing attacks on strong-password authentication protocol," *International Journal of Network Security*, vol.15, no. 1, pp. 64-67, 2013.

[14] I. E. Liao, C. C. Lee, and M. S. Hwang, "A password authentication scheme over insecure networks," *Journal of Computer and System Sciences*, vol. 72, no. 4, pp. 727-740, 2006.

[15] C. Lin, H. Sun and T. Hwang, "Attacks and solutions on strong-password authentication," *IEICE Transactions on Communications*, vol. E84-B, pp. 2622-2627, 2001.

[16] M. Peyravian and N. Zunic, "Methods for protecting password transmission," *Computers and Security*, vol. 19, no. 5, pp. 466-469, 2000.

[17] R. Ramasamy and A. P. Muniyandi, "An efficient password authentication scheme for smart card," *International Journal of Network Security*, vol. 14, no. 3, pp. 180-186, 2012.

[18] M. Sandirigama, A. Shimizu and M. T. Noda, "Simple and secure password authentication protocol (SAS),"

IEICE Transactions on Communications, vol. E83-B(6), pp. 1363-1365, 2000.

[19] R. Song, 'Advanced smart card based password authentication protocol," *Computer Standards and Interfaces*, vol. 32, pp. 321-325, 2010.

[20] S. K. Sood, "An improved and secure smart card based dynamic identity authentication protocol," *International Journal of Network Security*, vol. 14, no. 1, pp. 39-46, 2012.

[21] C. S. Tsai, C. C. Lee and M. S. Hwang, "Password authentication schemes: current status and key issues," *International Journal of Network Security*, vol. 3, no. 2, pp. 101-15, 2006.

[22] J. Xu, W. T. Zhu and D. G. Feng, "An improved smart card based password authentication scheme with provable security," *Computer Standards and Interfaces*, vol. 31, pp. 723-728, 2009.

[23] J. Yang and P. Shen, "A secure strong password authentication protocol," in *2010 2nd International Conference on Technology and Engineering* (*ICSTE*), vol. 2, pp. 355-357, 2010.

Xu Zhuang received his B.S. degree in Computer Science from Southwest Jiaotong University (SWJTU), Chengdu, China, in 2006, and is currently pursuing the Ph.D. degree in the Software Engineering Laboratory in Southwest Jiaotong University (SWJTU), Chengdu, China. His research interests include data mining, data hiding and information security.

Chin-Chen Chang received his Ph.D. degree in computer engineering from National Chiao Tung University. His first degree is Bachelor of Science in Applied Mathematics and master degree is Master of Science in computer and decision sciences. Both were awarded in National Tsing Hua University. Dr. Chang served in National Chung Cheng University from 1989 to 2005. His title is Chair Professor in Department of Information Engineering and Computer Science, Feng Chia University, from Feb. 2005. He is a Fellow of IEEE and a Fellow of IEE, UK. His research interests include database design, computer cryptography, image compression and data structures.

Zhi-Hui Wang received the BS degree in software engineering in 2004 from the North Eastern University, Shenyang, China. She received her MS degree in software engineering in 2007 and the PhD degree in software and theory of computer in 2010, both from the Dalian University of Technology, Dalian, China. Since November 2011, she has been a visiting scholar of University of Washington. Her current research interests include information hiding and image compression.

Yan Zhu received her B.S. and M.S. degrees in Computer Science from Southwest Jiaotong University (SWJTU), Chengdu, China, in 1986 and 1989, respectively. She received her Ph.D. degree in Computer Science from Darmstadt University of Technology, Germany in 2004. Yan Zhu is currently a professor of the School of Information Science and Technology, SWJTU and the director of the Laboratory of Software Engineering. Her research interests include data mining, Web information security, and Web spam detection.